ORACLE

MySQL 8.0 GIS機能チュートリアル

2020/01/25　OSC 2019 Osaka

Yoshiaki Yamasaki / 山﨑 由章
MySQL Principal Solution Engineer, Asia Pacific and Japan

Safe harbor statement

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ、お知らせ

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ、お知らせ

アップルップル

アプリケーション

「a-blog cms」は、アップルップルが開発する国産のCMSです。
様々な業種のWebサイト制作に使われており、2019年10月現在
公開可能事例550件、非公開事例3,400件以上の実績があります。

MySQLのGIS機能の活用例

「a-blog cms」では、投稿記事の属性として位置情報を含めることができます。位置情報は MySQLのgeometry型に保存されています。
この機能を活用することで、ユーザーの近くの情報を表示する、
といったWebページ制作を可能にしています。

アップルップル

位置情報を使用したWebページの例

名古屋地区の情報を発信しているSpyMasterでは、この機能を活用してユーザーの近くのスポット情報を表示できるようにしています。

ユーザーの現在地をJavaScriptのGeolocation APIを使って取得し、現在地からの距離をMySQLのST_Length関数で計算しています。

MySQLのGEOMETRY型とJavaScriptのGeolocation APIの活用事例(※)
https://speakerdeck.com/steelydylan/mysqlfalse-geometry-xing-tojavascriptfalse-geolocation-api-falsehuo-yong-shi-li

※本資料ではGLength関数を使った例が紹介されていますが、今後実装する際は
ST_ Length関数を使用下さい。
(GLength関数はMySQL 5.7で非推奨になり、MySQL 8.0で廃止されました).

SpyMaster (https://spymaster.jp)

GLength関数(ST_Length関数)で距離計算することで、現在地から近いスポット情報や、今表示しているスポットから近いスポットを表示

圓屋 (https://yenya.co.jp/locations)

GLength関数(ST_Length関数)で距離計算することで、現在地から近い順番に店舗情報を表示

ヤマレコ

アプリケーション

ヤマレコは「また山に行きたくなる」Webサービスです。登山の記録をヤマレコに残し、他の人と共有することが出来ます。登山者の知識・情報を共有することで登山計画を立てやすくし、遭難防止にも役立っています。また、登山者の位置情報をリアルタイムで共有することで家族が登山状況を確認出来たり、万が一の場合の救助活動にも役立ったりしています。

2005年10月にサービスを開始し、2019年9月時点で「月間140万人が訪問するWebサイト」、「40万ダウンロードの登山地図アプリ」となっています。

ヤマレコ

MySQLのGIS機能の活用例

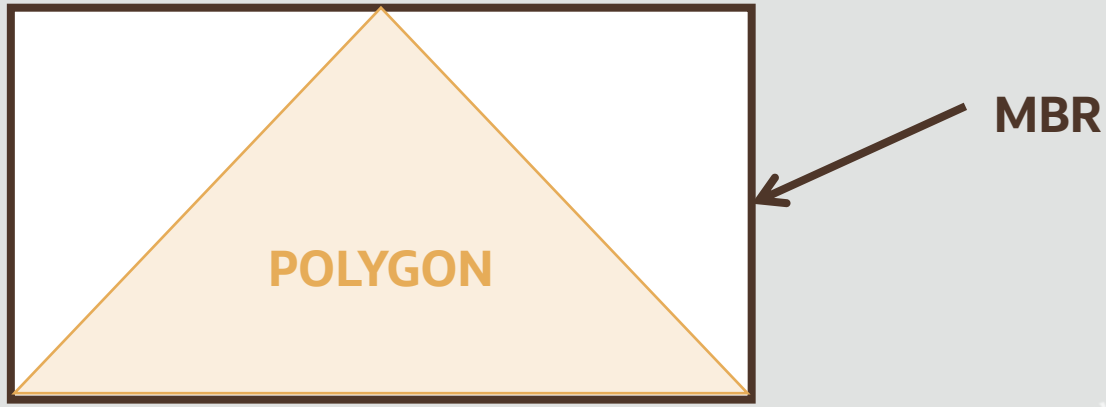
山行記録の地図検索機能をMySQLのGIS機能を活用して実装しています。空間インデックスを使用し、標準検索モードではMBRContains関数を、高精度検索モードではST_Contains関数を利用して、検索範囲に含まれる山行記録を高速に検索できるようにしています。

山行記録の地図検索機能

Leaflet、MySQL（Spatialインデックス＋Spatial関数）を使って実装

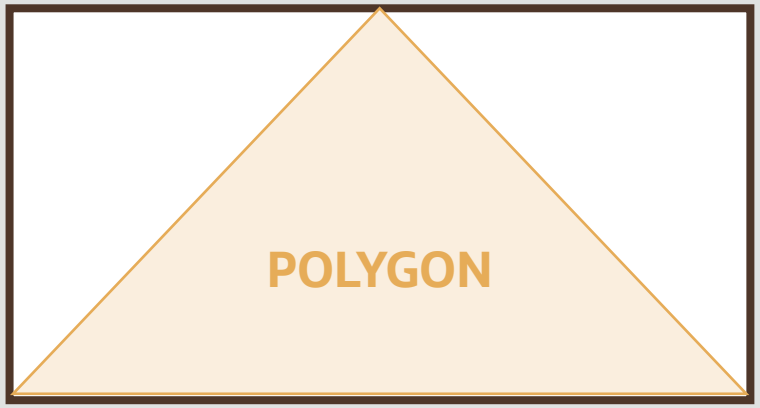
備考：MBRContains関数とST_Contains関数の違い

- MBR(Minimum Bounding Rectangles：最小外接矩形) とは？
  - その図形が隣接することのできる最小の矩形



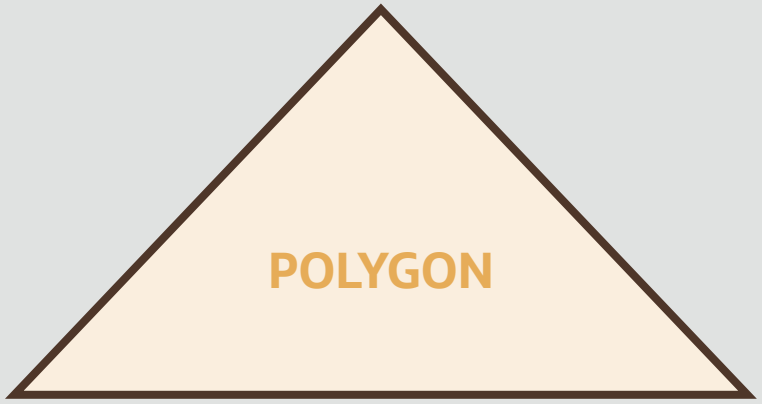
備考：MBRContains関数とST_Contains関数の違い

MBRContains関数



あるジオメトリデータが指定されたジオメトリデータのMBRの中に含まれているかどうかを判定する

ST_Contains関数



あるジオメトリデータが指定されたジオメトリデータの中に含まれているかどうかを判定する

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ、お知らせ

MySQLで扱えるデータ型

- POINT：点
- LINESTRING：線
- POLYGON：多角形

MySQLで扱えるデータ型

- POINT：点
  - 例：緯度、経度
- LINESTRING：線
  - 例：ルート（道筋）
- POLYGON：多角形
  - 例：市町村の区画

MySQLで扱えるデータ型

- POINT：点
  - 例：緯度、経度
- LINESTRING：線
  - 例：ルート（道筋）
- POLYGON：多角形
  - 例：市町村の区画

- GEOMETRY：POINT、LINESTRING、POLYGONをまとめて扱える

MySQLで扱えるデータ型

- 集合を扱えるデータ型
  - MULTIPOINT：POINTの集合
  - MULTILINESTRING：LINESTRINGの集合
  - MULTIPOLYGON：POLYGONの集合
  - GEOMETRYCOLLECTION： GEOMETRYの集合

MySQLのGEOMETRY型に関する補足

- PostGISでいうジオグラフィ型(※)に相当するものはない
  - SRID:4326のデータも扱うことはできるが、それだけに特化したデータ型は無い

※WGS84 地理座標系(SRID:4326)のみサポートするデータ型

空間データの表現方法

- WKT（Well-Known Text）
  - 幾何学オブジェクトをテキストで表現するための仕様
- WKB（Well-Known Binary）
  - 幾何学オブジェクトをバイナリで表現するための仕様
- MySQLの内部表現
  - WKBの先頭にSRIDを追加したもの

WKTの例

- POINT(15 20)　※区切りはスペース
- LINESTRING(0 0, 10 10, 20 25, 50 60)
- POLYGON((0 0,10 0,10 10,0 10,0 0))　※最初の点に戻る

- POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))
  ※中をくりぬくことも可能

WKTから空間データを生成する関数

- ST_GeomFromText()
- ST_PointFromText()
- ST_LineStringFromText()
- ST_PolygonFromText()
- ST_MultiPointFromText()
- ST_MultiLineStringFromText()
- ST_MultiPolygonFromText()
- ST_GeometryCollectionFromText()

WKBから空間データを生成する関数

- ST_GeomFromWKB()
- ST_PointFromWKB()
- ST_LineStringFromWKB()
- ST_PolygonFromWKB()
- ST_MultiPointFromWKB()
- ST_MultiLineStringFromWKB()
- ST_MultiPolygonFromWKB()
- ST_GeometryCollectionFromWKB()

データを変換する関数

- ST_AsText()
- ST_AsBinary()
- ST_SwapXY()

使用例：ST_GeomFromText、ST_AsTextの動作確認

```
mysql> SELECT ST_GeomFromText('LineString(1 1,2 2,3 3)');
+-------------------------------------------------------+
| ST_GeomFromText('LineString(1 1,2 2,3 3)')            |
+-------------------------------------------------------+
|                  ð?      ð?       @       @     @     @ |
+-------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ST_AsText(ST_GeomFromText('LineString(1 1,2 2,3 3)'));
+-----------------------------------------------------+
| ST_AsText(ST_GeomFromText('LineString(1 1,2 2,3 3)')) |
+-----------------------------------------------------+
| LINESTRING(1 1,2 2,3 3)                             |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

チュートリアル

- 1. GEOMETRYデータ(POINT、LINESTRING、POLYGON)の確認
  https://github.com/YoshiakiYamasaki/MySQL-GIS-Tutorial

- まずはSRIDを気にせずに、データの格納方法、参照方法を確かめましょう

チュートリアル

- 2. SRIDを指定してのデータ管理、Spatial関数(ST_Intersects)の確認
  https://github.com/YoshiakiYamasaki/MySQL-GIS-Tutorial

- SRIDを指定してテーブルを作成し、データを格納してみましょう
  - データ挿入時にもSRIDを指定する必要があります
  - 異なるSRIDのデータを挿入しようとすると、エラーになります
- Spatial関数を試してみましょう
  - 例として、ST_Intersects() 関数を使用しています

※ST_Intersects() 関数の場合はSRIDの指定は必須ではありませんが、ST_Distance()関数など、
SRIDを指定しないと動作しないSpatial関数もあります

MySQL Workbenchで確認したデータをOpenStreetMapで表示可能

- 手順は以下のQiitaの記事参照
  - MySQL 8.0にPOINTデータ(経度、緯度)を入れてMySQL Workbenchから検索し、OpenStreetMapで表示する
    https://qiita.com/yyamasaki1/items/c05f60357c69936fa0e7

ST_Distance() 関数：2地点間の距離を計算できる

構文：`ST_Distance(g1, g2 [, unit])`

※unitに指定できる単位は、以下のSQLで確認可能

```
SELECT * FROM INFORMATION_SCHEMA.ST_UNITS_OF_MEASURE;
```

チュートリアル

- 3. ST_Distance() 使用例
  https://github.com/YoshiakiYamasaki/MySQL-GIS-Tutorial

- 2地点の位置情報をテーブルに格納し、ST_Distance () 関数で距離計算してみましょう
  - 例として弊社の本社と赤坂オフィスの位置情報を格納し、距離計算しています

ST_Distance() 使用例

```
mysql> CREATE TABLE test(id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(30),
         location POINT SRID 4326);
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO test(name, location) VALUES('Oracle Aoyama Center',
         ST_GeomFromText('POINT(35.67133 139.71857)', 4326));
Query OK, 1 row affected (0.01 sec)

mysql> SELECT id, name, ST_ASTEXT(location) FROM test;
+----+----------------------+-----------------------------+
| id | name                 | ST_ASTEXT(location)         |
+----+----------------------+-----------------------------+
|  1 | Oracle Aoyama Center | POINT(35.672238 139.718664) |
+----+----------------------+-----------------------------+
1 row in set (0.00 sec)
```

SRID:4326のAxis OrderはLat-Long
なので、緯度-経度の順番で指定

ST_Distance() 使用例

```
mysql> INSERT INTO test(name, location) VALUES('Akasaka Center Building',
         ST_GeomFromText('POINT(35.67686 139.73366)', 4326));

Query OK, 1 row affected (0.01 sec)

mysql> SELECT ST_Distance((SELECT location FROM test WHERE id=1),
         (SELECT location FROM test WHERE id=2), 'metre')/1000 AS Km FROM dual;
+--------------------+
| Km                 |
+--------------------+
| 1.497605951422511 |
+--------------------+
1 row in set (0.00 sec)
```

備考：データINSERT時にAxis Orderを明示的に指定可能

```
mysql> INSERT INTO test(name, location) VALUES('Oracle Aoyama Center',
          ST_GeomFromText('POINT(139.71857 35.67133)', 4326,
          'axis-order=long-lat'));
Query OK, 1 row affected (0.01 sec)
```

※axis-orderオプションは、地理座標系の場合のみ指定可能

備考：投影座標系でのaxis-orderオプションに関して

- 投影座標系でもaxis-orderオプションが必要になるケースがあると考えていますが、現状開発チームは必要性を認識していません
  - 「投影座標系の場合、軸の順番に本質的には意味がないため不要」、という考え方
  - とはいえ、MySQLはあくまでデータベースなので、入力に使うツール/アプリケーションと出力に使うツール/アプリケーションのaxis-orderが異なっていると、問題になる
- 以下の機能追加リクエストを登録していますので、是非Affects meして下さい！！
  - Bug#97221: Enabling the axis-order option even in Cartesian coordinate systems https://bugs.mysql.com/bug.php?id=97221

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ、お知らせ

外部ファイルからMySQLへのデータ格納方法　その1

- シェープファイル、GeoJSONファイルのデータを取り込む方法について、先日のFOSS4G Niigataでの発表資料にまとめています
  - MySQL 8.0で強化されたGIS機能と使用事例のご紹介＋α
    https://speakerdeck.com/yoshiakiyamasaki/mysql-8-dot-0deqiang-hua-saretagisji-neng-toshi-yong-shi-li-falsegoshao-jie-a

備考：ogr2ogr(GDAL)のインストール方法に関する解説記事

- MySQLに対応したogr2ogr(GDAL)をインストールする方法まとめ
  https://qiita.com/miyauchi/items/87921eb3ad630db1624a

備考：シェープファイルのimport/exportに関して

- シェープファイルを扱うためにGDALが必要になるのは不便なので、MySQL製のシェープファイルのimport/exportツールが欲しいと考えています
- 以下の機能追加リクエストを登録していますので、是非Affects meして下さい！！
  - Bug#90023: [Feature request] Shape File import/export tool https://bugs.mysql.com/bug.php?id=90023

外部ファイルからMySQLへのデータ格納方法　その2

- シェープファイルをMySQLに取り込むためのSQL文を生成できるshap2mysqlというツールが存在します
  - MySQLにシェープファイルをインポートするツール(shp2mysql)を作った
    https://qiita.com/miyauchi/items/b4e810b3becf2cf07e2f
  - GitHubのページ
    https://github.com/hajime-miyauchi/shp2mysql
  - MySQLに世界地図をインポートする　※shp2mysqlを使って世界地図をインポートする方法の解説記事
    https://qiita.com/miyauchi/items/4e5d77f7b49f4a3fe11a

MySQLのGIS機能を手軽に試してもらうために

- (会社としての公式な活動では無く)私の個人的なプロジェクトとして、MySQLに取り込み済みのGISデータの配布を始めました
- 手始めに、都道府県の境界データ(※)を以下で配布しています
  - https://github.com/YoshiakiYamasaki/MySQL-GIS-Data-Japan-eStat
- 47都道府県のデータを配布中

※出典：政府統計の総合窓口(e-Stat)（https://www.e-stat.go.jp/）
(e-Statからダウンロードした世界測地系のシェープファイルをMySQLに取り込み、
SHAPE列にSRIDとSpatialインデックスを追加したものを配布)

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ、お知らせ

GeoJSONを扱えるSpatial関数もあります

- ST_AsGeoJSON()：ジオメトリ型のデータを入力し、GeoJSONデータを出力
- ST_GeomFromGeoJSON()：GeoJSONデータを入力し、ジオメトリ型のデータを出力

※詳細を解説しているマニュアル
MySQL 8.0 Reference Manual / 12.15.11 Spatial GeoJSON Functions
https://dev.mysql.com/doc/refman/8.0/en/spatial-geojson-functions.html

GeoHashを扱えるSpatial関数もあります

- ST_GeoHash()：経度、緯度(POINT型のデータでも可)を入力し、GeoHashを出力
- ST_LatFromGeoHash()：GeoHashを入力し、経度を出力
- ST_LongFromGeoHash()：GeoHashを入力し、緯度を出力
- ST_PointFromGeoHash()：GeoHashを入力し、POINT型のデータを出力

※SRIDが4326(もしくは0)の場合のみ使用可能

※詳細を解説しているマニュアル
MySQL 8.0 Reference Manual / 12.16.10 Spatial Geohash Functions
https://dev.mysql.com/doc/refman/8.0/en/spatial-geohash-functions.html

GeoHash以外のハッシュ化手法に関する機能追加リクエストも提出しています

- Bug#96759：Add ST_Quadkey(),ST_PointFromQuadkey(),ST_Lat/LongFromQuadkey() function https://bugs.mysql.com/bug.php?id=96759
- Bug#96760：Add ST_LocaPoint(),ST_PointFromLocaPoint(),ST_Lat/LongFromLocaPoint() function https://bugs.mysql.com/bug.php?id=96760
- Bug#96761：Add ST_GeoHex(),ST_PolygonFromGeoHex() function https://bugs.mysql.com/bug.php?id=96761
- Bug#96762：Add ST_Pluscodes(),ST_PointFromPluscodes(),ST_Lat/LongFromPluscodes() function https://bugs.mysql.com/bug.php?id=96762

※「Affects me」大歓迎です！！

GeoHash以外のハッシュ化手法の特徴

- Quadkey
  - メッシュが正方形、メッシュ上位と下位の分割が4分木であるため、使いやすい（GeoHashはメッシュが長方形、メッシュ上位と下位の分割が32分木）
- LocaPoint
  - 人間が認識しやすいコード体系（英文字・英文字・数字×4回）
- GeoHex
  - メッシュが六角形　⇒　隣接するメッシュへの距離が等距離になる
  - 六角形の見た目がカッコイイ！（エンターテイメント向き！？）
- plus+codes
  - 人間が認識しやすいコード体系（region code + city code + neighbourhood code + building code ）
  - Googleのエンジニアが開発、Google Mapでもサポートされている

注意事項

- ST_Intersects()、ST_Overlaps()の実行速度が非常に遅いという問題が発生していて、現在調査中です
  - Bug#96311: ST_Intersects() is very slow on MySQL 8.0
    https://bugs.mysql.com/bug.php?id=96311

- ST_Within()で代替できるケースでは、現状はST_Within()を使用下さい
  (ST_Within()は高速に実行できています)

アジェンダ

MySQLのGIS機能を使用したシステムの例

MySQLで扱えるデータ型、データの挿入/参照方法

外部ファイルからMySQLへのデータ取り込み方法

その他のトピック

まとめ

まとめ

- MySQLのGIS機能を是非試してみて下さい！
- 試してみて気づいたことがあれば、是非フィードバック下さい！！
  - MySQL Bugs

    https://bugs.mysql.com/